Flow-level network models: have we reached the limits?

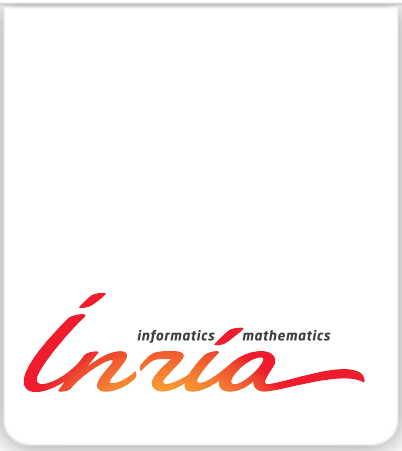
Pedro Velho, Lucas Schnorr, Henri Casanova, Arnaud Legrand

Flow-level network models: have we reached the limits?

Pedro Velho, Lucas Mello Schnorr, Henri Casanova, Arnaud Legrand

RESEARCH REPORT
N° 7821
November 2011
Project-Team MESCAL

ISSN 0249-6399 ISRN INRIA/RR--7821--FR+ENG



Flow-level network models: have we reached the limits?

Pedro Velho, Lucas Mello Schnorr, Henri Casanova, Arnaud Legrand

Project-Team MESCAL

Research Report n° 7821 — November 2011 — 24 pages

Abstract: Researchers in the area of distributed computing conduct many of their experiments in simulation. While *packet-level* simulation is often used to study network protocols, it can be too costly to simulate network communications for large-scale systems and applications. The alternative is to simulate the network based on less costly *flow-level* models. Surprisingly, in the literature, validation of these flow-level models is at best a mere verification for a few simple cases. Consequently, although distributed computing simulators are widely used, their ability to produce scientifically meaningful results is in doubt. In this work we focus on the validation of state-of-the-art flow-level network models of TCP communication, via comparison to packet-level simulation. While it is straightforward to show cases in which previously proposed models lead to good results, instead we systematically seek cases that lead to invalid results. Careful analysis of these cases reveal fundamental flaws and also suggest improvements. One contribution of this work is that these improvements lead to a new model that, while far from being perfect, improves upon all previously proposed models. A more important contribution, perhaps, is provided by the pitfalls and unexpected behaviors encountered in this work, leading to a number of enlightening lessons. In particular, this work shows that model validation cannot be achieved solely by exhibiting (possibly many) "good cases." Confidence in the quality of a model can only be strengthened through an invalidation approach that attempts to prove the model wrong.

Key-words: Simulation, fluid network models, validity, methodology

---

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Modèles réseaux fluides: avons-nous atteint leur limite?

Résumé : Dans le domaine du calcul distribué, les chercheurs réalisent un grand nombre de leurs expériences à l'aide de simulations. Si les simulations de niveau paquet ont été intensément utilisées pour étudier des protocoles réseaux, elles sont en général trop coûteuses pour simuler les communications de systèmes et d'applications distribués à grande échelle. Une alternative classique consiste à simuler le réseau à l'aide de modèles fluides moins coûteux. Curieusement, dans la littérature, la validation de ces modèles se réduit en général au mieux à une simple vérification sur quelques cas simples. Par conséquent, bien que ces simulateurs de systèmes de calcul distribué soient largement utilisés, leur capacité à produire des résultats pertinents sur le plan scientifique est douteuse. Dans cet article, nous nous intéressons à la validation de modèles fluides de TCP reconnus dans la littérature en les comparant à des simulations de niveau paquet. Puisqu'il est aisé de montrer des cas dans lesquels ces modèles produisent de bons résultats, nous nous intéressons systématiquement à la recherche de cas invalidant ces modèles, une analyse méticuleuse de ces situations révélant les failles profondes de ces modèles et suggérant parfois des améliorations possibles. Une des contributions de cette étude est que ces améliorations nous ont amené à la proposition d'un modèle qui, bien qu'imparfait, produit des résultats de meilleur qualité que les modèles ayant été précédemment proposés. Une contribution probablement plus importante réside dans la mise en lumière des difficultés auxquelles nous nous sommes heurtés et qui ont conduit à un certain nombre de leçons. En particulier, ce travail montre que la validation d'un modèle ne s'effectue pas en exhibant des situations (même en nombre) où tout se passe bien. La confiance en la qualité d'un modèle ne peut être renforcée qu'en accumulant les tentatives infructueuses d'invalidation.

Mots-clés : Simulation, modélisation fluide du réseau, validité, méthodologie

1 Introduction

A large part of distributed computing research is empirical in nature, requiring that experimental results are obtained to evaluate and compare proposed approaches. Conducting experiments in (large-scale) distributed systems, if at all possible, is usually time-consuming and labor-intensive. Distributed systems are subject to software and hardware heterogeneity which complicates application deployment. Intermittent and non-controllable load variations and failures typically preclude reproducible experiments. Furthermore, systems may not be available to the purpose of research experiments that could disrupt production use. Even if a fully controllable and available system is available, the researcher may want to study systems that are not necessarily available (e.g., because they are being deployed, to explore "what if" scenarios). For these reasons, many published works in distributed computing rely on simulation.

Most distributed computing simulators provide simulation models for compute components (e.g., servers) and network components (e.g., routers, links, network cards). Models that capture the operation of these components in detail (e.g., cycle-level simulation of a processor, packet-level simulation of a router) prove intractable when simulating large-scale systems with many such components. As a result, simulation models must trade off a higher level of detail for a higher compute speed. Thus rises the question of model validity: to which extent do these trade-offs reduce the accuracy of obtained simulation results and can these results be used to draw scientifically valid conclusions? In this work we study this question in the context of network simulation models, many of which have been proposed in the literature.

Although using accurate models seems paramount, many distributed computing simulators use simplistic network models that have not been validated [2, 4, 31]. When validation is attempted, it is often done only for a few cases in which the model is expected to work well [40, 41], thereby merely verifying that the model implementation is correct and that its results are not completely unreasonable. While commonplace in computer science, such loose methodology is unacceptable in other fields. In the natural sciences, for instance, one cannot fully prove the correctness of a model: a model is considered valid only until its invalidity is demonstrated. Model validation is thus an important component of the research activity, and it goes through so-called invalidation studies that explore a wide range of scenarios for which the model's behavior is either unknown or expected to be invalid. Through these invalidation studies, the model is improved or refuted, leading to increasingly precise knowledge of the range of situations in which the model's result are to be considered meaningful (or meaningless).

In this work, we focus on flow-level network models of the Transmission Control Protocol (TCP) to be used in distributed computing on wide-area networks. The goal of these models is to capture complex network behavior using tractable mathematical models, i.e., that can be computed quickly. Several flow-level models of TCP have been proposed in the literature [22, 23, 24], and we ourselves have proposed such a model [38], which is used in the SIMGRID simulation framework [5]. Our contributions in this work are as follows:

- We obtain new insight into flow-level modeling using a systematic validation approach based on invalidation experiments.
- This approach invalidates the models in [22, 23, 24].
- This approach also suggests several improvements to the model recently described in [38].

- The improved model, while far from being perfect, is, to the best of our knowledge, the most accurate flow-level TCP model available to date.
- Our approach, and the difficulties encountered, provide important (and sobering) lessons for model development and validation in the context of distributed computing simulation.

The rest of this paper is organized as follows. Section 2 explains our general approach for model development validation, which is inspired by the critical method used in the natural sciences. Section 3 discusses related and details directly relevant previous work. Section 4 presents our invalidation study and the improvements it suggests for flow-level models. Section 5 discusses the limits of flow-level modeling, as highlighted by our invalidation study. Finally, Section 6 summarizes our contributions and outlines future research directions.

2 Methodological Foreword

In this section, we recall the principles of the critical method used in most natural sciences, and make a case for using it in computer science. We make recommendations for the sound evaluation of computer system models, which provide a methodological framework that distinguishes this work from most previous works in this area.

2.1 The Critical Method

Humans are driven to look for patterns in nature, with the innate conviction that natural *laws* explain the world around us. Physics, chemistry and biology are sciences that seek to discover such laws. It is tempting to reason that a law, observed to be true in all situations experienced so far, will be true in other situations and at later times (this is called *inductive* reasoning). Unfortunately, as pointed out by David Hume centuries ago, it is not possible to establish natural laws based on observations. And indeed, human history is rife with long established laws that were suddenly shown to be false, often with dramatic implications for human knowledge. Nevertheless, as *pragmatists*, we build knowledge about the world around us as a way to better the human condition. There is a pragmatic justification for seeking true laws of nature, whether they exist or not, and to build theories even if we cannot prove them true. Consequently, as well recognized by Popper [30], theories may be *refuted* by new observations at any time (e.g., Newton's theory was refuted by Einstein's relativity theory). Theory refutation is accepted as a fundamental part of the life cycle of science and the "theoretician" strives to develop non-refuted theories, in the hope that some of them may be true.

The approach for invalidating a target yet-to-be-refuted theory is to build a *falsifying law*, i.e., a law whose universality is sufficiently low that it does not subsume the target theory. The hope is that the falsifying law will suggest a *crucial experiment*, i.e., an experiment that would invalidate either the target theory or the falsifying law. Using this *critical method*, perhaps all theories end up being refuted and that no theory can be developed that explains observations. If, instead, several yet-to-be-refuted theories are available, one should prefer the ones that have been tested the most and/or the ones with the higher level of information and of explanation (i.e., the most universal). A new theory should both explain observations that were also explained by the refuted theory, and explain why the refuted theory failed to explain observations now explained by the

new theory. Finally, theories should not be *ad hoc* in the sense that they should not be developed specifically for the finite set of situations that they are to be tested upon.

2.2 Computer World vs. Natural World

Unlike natural systems, computer systems are built by humans following specific designs. As such, they ought to follow pre-determined laws. In particular, these systems should behave deterministically and thus be amenable to inductive reasoning. One may argue that computers systems are subject to natural phenomena (e.g., a cosmic ray may corrupt memory). Although such phenomena are studied, most computer science research assumes that the laws of nature under which computers are built are forever true. Given this hypothesis and the determinism of closed computer systems, and by contrast with natural systems, it is possible to establish true computer laws.

Known such true laws are often "low-level" in that they govern elementary operations (e.g., an ALU can add two integers in one cycle). It is expected that some laws also govern the "high-level" behavior of computer systems. Discovering and understanding these high-level laws is paramount for designing better computer systems. However, computer systems, albeit human-made, have reached such complexity and scale that deriving true high-level laws is often intractable. A part of computer science is thus devoted to developing theories that approximate the high-level behavior of computer systems. We call such approximations *models*, which are the bases for simulation (i.e., the implementation of a model through a computer program).

Two approaches are used to build models of a computer system: from observations of the full system (*top-down* approach) or from analysis of low-level computer laws (*bottom-up* approach). The former is akin to the approach used in natural sciences, while the latter is not possible in natural sciences except to strengthen our confidence in theories describing the world at different scales (e.g., the macroscopic laws of thermodynamics are consistent with the microscopic laws of thermodynamics).

Unfortunately, even the bottom-up approach does not guarantee that truthful high-level models can be obtained. To make construction tractable, such models are built while approximating or neglecting certain aspects of the system. In fact, models are often built to explicitly trade off realism and universality for simplicity and tractability. Consequently, high-level computer models, just like theories about the natural world, must be evaluated until they are refuted and replaced by better models.

2.3 Modeling in Computer Science

Using the critical method in the area of computer system modeling has the following implications:

1. Inductive reasoning should be banned: *the quality of a model cannot be proven solely by accumulating observed situations in which the model is effective*.

2. Instead, and perhaps counter-intuitively, *the quality of a model should be proved by searching for situations in which the model is not effective*. Model refutation through falsifying laws and crucial experiments is the way to improve our knowledge of computer systems.

3. A newly proposed model should, if at all possible, explain successes and failures of previously proposed and now refuted model. As a consequence, "magical" model parameters that do not have clear significance but make it possible to "fit"

a model to a set of observations should be avoided. The objective is *to distinguish between true improvement of knowledge and ad hoc models*.

Following these principles, several candidate yet-to-be-refuted models may be available. Two additional practical considerations must be taken into account for selecting which model to use: *complexity* and *instantiability*. One should prefer the model with the best accuracy over a set of relevant situations, or the larger set of situations in which accuracy is above some threshold. Furthermore, everything else being equal, a model with lower complexity (i.e., less computationally intensive) should be preferred. One may, however, knowingly opt for a less complex model over a more accurate model solely for the sake of computational tractability. Instantiability is the ability to determine sound values for the model's input parameters. When constructing a model, it is often tempting to increase the number of input parameters. The result, however, may be a model of little practical significance as the instantiation of each parameter could requires a whole set of complex experimental measurements (which often compels users to use unsound "guesses" as parameter values).

With all this in mind, we can now take a more scientifically grounded approach to computer system modeling research. In particular, we apply this approach to models previously proposed by other researchers and to our own for the purpose of network modeling and simulation. Through crucial experiments we can refute certain models and to explain their shortcomings. More fundamentally, our goal is to measure and understand the inherent limitations of an entire class of network modeling techniques.

3 Related and Previous Work

3.1 Packet-Level Simulation

Packet-level network simulations are discrete-event simulations with events for packet emission or reception as well as network protocol events. The simulation can reproduce the movements of all network packets and the behavior of the whole TCP stack down to the IP level. Packet-level simulations have been widely used for studying fine-grained properties of network protocols. The most popular packet-level simulator is NS2 [18], while more recent simulators include NS3 [28], GTNetS [32] and OMNet++ [37]. The TCP stack models found in simulators such as GTNets or NS2 are simplified versions of the TCP stack. More recent developments [20] allow the use of real TCP stack implementations, which is slower but more realistic (i.e., real TCP stack implementations might have features/bugs that are absent from simplified versions used exclusively for simulation purposes).

Except maybe for wireless networks where the modeling of the physical layer is challenging, packet-level simulation is generally recognized by the network community as trustworthy, and it serves as reference for network protocol experiments. Unfortunately, it can lead to long simulation times [16] since the life cycle of each packet is simulated through all protocol layers all the way to a simulated physical layer. Packlevel simulation is thus not sufficiently scalable for simulating applications that send substantial amounts of data through the network. Fortunately, the level of detail provided by packet-level simulation is likely unnecessary for studying such applications. As a possible solution, simulators in the area of grid computing (e.g., GridSim [4], early versions of SimGrid [5]), have attempted widely simplified packet-level simulation (e.g., wormhole routing, no implementation of any network protocol). These simulators are easily shown to produce nonsensical simulated communication times

even in simple cases. Besides, these solutions also face scalability issues as more realistic packet-level simulators. To ensure scalability, radically more coarse-grain network models are required.

3.2 Flow-level Simulation

In packet-level simulation, all the packets that comprise a network communication between two endpoints are simulated individually. In flow-level simulation, which is used by simulators in various domains [2, 7, 29, 40], the communication, or *flow*, is simulated as a single entity. In this case, the simulation amounts to solving a bandwidth sharing problem: Determine how much bandwidth is allocated to each flow. More formally:

> *Consider a connected network that consists of a set of links $\mathcal{L}$, in which each link l has capacity B_l. Consider a set of flows $\mathcal{F}$, where each flow is a communication between two network vertices along a given path. Determine a "realistic" bandwidth allocation ϱ_f for flow f, so that:*
>
> $$\forall l \in \mathcal{L}, \sum_{f \text{ going through } l} \varrho_f \leqslant B_l \ . \tag{1}$$

Given the computed bandwidth allocation (which defines all data transfer rates), and the size of the data to be transmitted by each flow, one can determine which flow will complete first. Upon completion of a flow, or upon arrival of a new flow, the bandwidth allocation can be reevaluated. Usually, this reevaluation is memory-less and does not depend on past bandwidth allocations. This approach makes it possible to quickly step forward through (simulated) time, and thus is attractive for implementing scalable simulations of large-scale distributed systems with potentially large amounts of communicated data. However, it is only an approximation as it ignores phenomena such as protocol oscillations or slow start (even though slow start can sometimes be modeled via an additional communication latency that depends on message size [11]). Perhaps more crucially, the whole approach is precondition on computing a bandwidth sharing solution that corresponds to the bandwidth sharing behavior of real-world network protocols, and in particular of TCP.

The TCP network protocol is known for its lack of stability, due to its additive increase multiplicative decrease window size policy. Nevertheless, its average behavior has been characterized as the solution of an optimization problem. By making an analogy between the equations governing expected window size that follow from TCP and a distributed gradient algorithm, Low *et al.* [22] have proved that the steady-state throughput of network flows are similar to those obtained by solving a global optimization problem under constraints (1). We briefly recall the main principle of this modeling approach by illustrating its use for the Reno protocol using RED as a queue policy for routers.

Let $w_f(t)$ be the window size of the emitter of flow f at time t. $w_f(t)$ is thus the number of packets of f for which no ack has yet been received at time t. Let d_f be the equilibrium round trip time (propagation plus equilibrium queuing delay) of f, which is assumed to be constant. The instantaneous data transfer rate $\varrho_f(t)$ is thus equal to $w_f(t)/d_f$. Using the RED protocol, if we denote by p_l the loss probability on link l, the probability of packet loss for flow f is:

$$q_f = 1 - \prod_{l \text{ traversed by } f} (1 - p_l) \approx \sum_{l \text{ traversed by } f} p_l \quad \text{when } p_l \ll 1$$

At time t, f's emitter transmits $\varrho_f(t)$ packets per time unit, and receives (positive and negative) acknowledgments at approximately the same rate, assuming that every packet is acknowledged. On average, f's emitter receives $\varrho_f(t)(1 - q_f(t))$ positive acknowledgments per time unit and each positive acknowledgment increases the window $w_f(t)$ by $1/w_f(t)$ (additive increase). It also receives, on average, $\varrho_f(t)q_f(t)$ negative acknowledgments (marks) per time unit, halving the window for each. The net change to the window between two packet emission, which occurs roughly every $\delta_f(t) = d_f/\varrho_f(t)$ time units, is then obtained as:

$$w_f(t + \delta_f(t)) - w_f(t) = \frac{1}{w_f(t)}(1 - q_f(t)) - \frac{w_f(t)}{2}.q_f(t) \ .$$

Hence, $\varrho_f(t)$ follows the following ODE:

$$\frac{d\varrho_f}{dt} = \frac{1 - q_f(t)}{d_f^2} - \frac{1}{2}q_f(t)\varrho_f(t)^2 \ .$$

The value $q_f(t)$ accounts for the contention along f's path and for the fact that the capacity of every link should not be exceeded (i.e., that constraints (1) are enforced). Associating a Lyapunov function to this ODE makes is possible to prove that the trajectories converge to a point that maximizes

$$\sum_{f\in\mathcal{F}} \frac{\sqrt{2}}{d_f} \arctan\left(\frac{d_f\varrho_f}{\sqrt{2}}\right) \text{ under constraints (1).} \tag{2}$$

Similarly, it can be proven that TCP Vegas achieves some form of weighted proportional fairness [22], i.e., it maximizes the weighted sum of the logarithm of the throughput of each flow:

$$\sum_{f\in\mathcal{F}} d_f \log(\varrho_f) \quad \text{under constraints (1).} \tag{3}$$

These approaches are *bottom-up*, as defined in Section 2.2, and are thus attractive because they capture the specifics of the underlying network protocol. Authors have also proposed *top-down* approaches, that are not derived from an analysis of the TCP protocol. A commonly used bandwidth sharing objective is Max-min fairness. This sharing is obtained by recursively maximizing

$$\min_{f\in\mathcal{F}} w_f\varrho_f \quad \text{under constraints (1),} \tag{4}$$

where w_f is generally chosen as the round-trip time of flow f. There are two rationales for this objective. First, it corresponds to what many would naïvely expect from a network, i.e., be "as fair as possible" so that the least favored flows receive as much bandwidth as possible while accounting through weights w_f for the well-known RTT-unfairness of TCP. Second, there is a simple algorithm for solving the optimization problem [3], whereas solving non linear problems (such as the ones involved with $\arctan$-based functions or weighted proportional fairness) requires more elaborate techniques. Previous studies have shown that Max-min fairness does not exactly correspond to bandwidth sharing under TCP but that it is a reasonable approximation in many relevant cases [8, 10].

3.3 Accuracy of Flow-level Simulation

The key question that we study in this work is whether flow-level simulation can provide accurate results. Some simulators (e.g., OptorSim [2], GroudSim [29]) do implement flow-level models. These models are easily shown to be inaccurate because based on broken implementations of Max-min bandwidth sharing. Besides such blatantly invalid models, there is a striking dearth of validation studies in the literature. Those works that do propose flow-level models [23, 22] are driven by protocol design goals [24], and as such they merely present a few test cases to illustrate the correctness of the models. Unfortunately, as explained in Section 2.3, demonstrating validity by showcasing (a few) instances in which a model works, whether bottom-up or top-down, is not sufficient. More in-depth validation studies were conducted by Marchal *et al.* [8, 6] and by Fujiwara and Casanova [16]. These studies explore a moderate range of experimental scenarios, but mostly they end up exhibiting instances in which the evaluated models work reasonably well. None of these works follow the critical approach described in Section 2.

Evaluating the validity of flow-level models in complex and diverse scenarios raises practical difficulties, such as requiring that real-world networks be configured for each scenario of interest. One convenient solution is to compare results obtained with flow-level models to results obtained using packet-level simulation. This approach raises the question of whether packet-level simulation is representative of real-world networks. Answering this question is out of the scope of this work. However, based on the confidence placed by the network community in its packet-level simulators, it seems reasonable to declare packet-level simulation the ground truth. The work in [16] provides an evaluation of the flow-level model implemented in the SimGrid simulation framework at that time. Conveniently, SimGrid provides an interface to the GTNetS packet-level simulator, which greatly eased the comparison of flow-level and packet-level results. GTNetS is no longer officially supported and the latest version of SimGrid also provides an interface to NS3. (Both GTNetS and NS3 implement the same TCP stack.) The current version of SimGrid implements flow-level models based either on Max-min fairness or on Low *et al.*'s work [22].

These capabilities of SimGrid make it a convenient framework for studying the validity of flow-level models. Our recent work in [38], on which this work builds, has taken advantage of these capabilities to evaluate and improve upon a top-down Max-min flow-level model. In that work the following advances were made:

- **Linearity:** The communication time of a message for flow f is given by $t_f = S_f/\varrho_f + L_f$ where S_f is the message size, ϱ_f is the bandwidth allotted to f, and L_f is the sum of the latencies of the links traversed by f. However, L_f and B_l (used in the computation of ϱ_f) are physical characteristics that are not directly representative of what may achieved by flows in practice. The protocol overhead should be accounted for, which can be done by multiplying all latencies by a factor $\alpha > 1$ and all bandwidths by a factor $\beta < 1$. α can account for TCP slow-start and stabilization, which prevent flows from instantaneously reaching steady-state. β can account for packing and control overheads. This simple change leads to a good approximation for a single flow on a single link when message size is larger than 100KB (with $\alpha = 10.2$ and $\beta = 0.92$).

- **Flow Control Limitation:** TCP's flow control mechanism is known to prevent full bandwidth usage as flows may be limited by large latencies [26, 14, 19]. This well-known phenomenon can be captured in a flow-level model by adding, for

each flow, the constraint that:

$$\varrho_f \leqslant W_{\max}/(2.L_f) \ , \tag{5}$$

where $W_{\max}$ is the configured maximum window size. The validity of this enhancement is demonstrated for a single flow going through a single link.

- **Bottleneck Sharing:** TCP protocols are known to be RTT-unfair, hence when two flows contend for bandwidth on a bottleneck link, they are assigned bandwidth inversely proportional to their round trip times [25]. On a simple dogbone topology, but for a wide range of bandwidth and latency parameters, this round trip time is well approximated by the following formula:

$$RTT_f = \sum_{l \text{ traversed by } f} L_l + \frac{\gamma}{B_l} \ , \tag{6}$$

where γ is a fixed value for all flows ([38] shows that $\gamma = 8,775$ provides a good approximation).

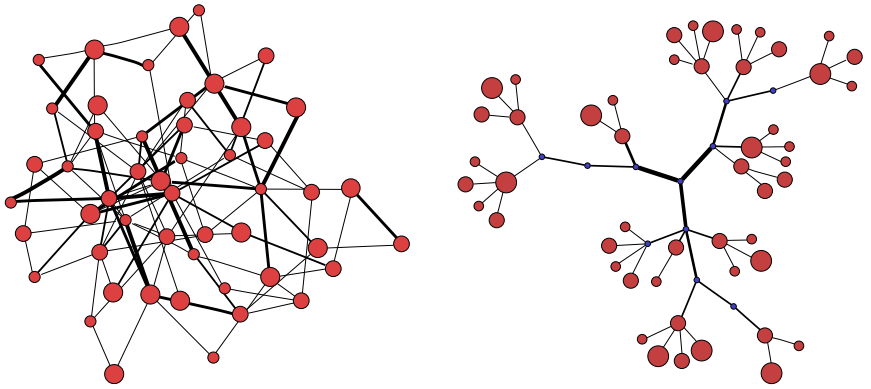
Although they may look *ad hoc*, these model enhancements do have sound justifications and parameter values have natural interpretations. Improvements were demonstrated on dozens of randomly generated networks with 50 to 200 nodes and 150 flows with randomly chosen sources and destinations. Yet, in spite of good average results, occasionally a flow has a throughput that is more than a factor 20 away from that obtained using packet-level simulation.

4 On the (In)validation path

In this section, we build on the work in [38], applying the critical method outlined in Section 2. Interestingly, the validity of the bandwidth sharing model itself is not questioned in [38]. In fact, both Max-min sharing and arctan-based sharing (based on Low *et al.* [22]) lead to the exact same bandwidth allocation on simple cases like dogbone platforms, which is easily formally provable. Interestingly, although not reported in [38] because unexplained, the use of $\arctan$-based sharing instead of Max-min sharing did not seem to provide any significant improvement for more complex topologies. In other words, the main sources of error seem to be model instantiation errors rather than fundamental flaws of the bandwidth sharing model. This is surprisingly given that in the literature there is an unstated assumption that the bandwidth sharing model itself is of critical importance. In what follows, we describe our critical method approach to understand, and hopefully resolve, the errors of the flow-level model developed in [38]. We follow the same approach of comparing results obtained with this model and with packet-level simulation, configuring packet-level simulators to implement TCP Vegas. We mostly report on packet-level simulation results obtained with GTNeTS. When results seemed surprising we confirmed them using NS3 to ensure that results were not due to artifacts of the packet-level simulator.

4.1 Phase Effects

As described in Section 3.3, [38] has proposed and evaluated a (top-down) flow-level model based on Max-min optimization. For completeness, we recall here the evaluation methodology used in [38], which we also use in this work, as well as the final results



(a) A BRITE platform: scenarios based on this kind of platform have little contention.

(b) A Tiers platform: hierarchy leads to more contention.

Figure 1: Two typical 50 nodes topologies used in our study.

of this evaluation. Four sets of 10 random topologies were generated with the Waxman model [39] and the BRITE generator [27] (Figure 1(a) illustrates a typical 50-node random topology). Although there has been a long-standing debate [9, 21, 35] on network topology characteristics and the existence of power laws [13], we use these two generators because at small scale they are probably more meaningful than degree-based generators [1]. These sets comprise either small (50 nodes) or large (200 nodes) and either mostly homogeneous (B_l follows a uniform distribution in $[100, 128]$ MB/s) or heterogeneous (B_l follows a uniform distribution in $[10, 128]$ MB/s) topologies. Network link latencies are computed by BRITE based on the Euclidean distance and are in the interval $(0, 5]$ms. 100 flows are generated between random pairs of end-points and 10 different sets of such random flows are simulated for each network configuration. For the experiment described hereafter, all flows transfer 100MB and the maximum TCP window size is set to 64 KiB. Overall, there are 160 experimental scenarios[1].

In the simulation, all flows start at exactly the same time. The simulation ends as soon as one flow completes, and the amount of data transferred by each flow is determined. The rationale for stopping the simulation after the first completion is to avoid computing data transfer rates over a time period in which a variable number of flows are active. Based on the amounts of data transferred, the bandwidth allocated to each flow is then computed. This enables to focus on instantaneous bandwidth sharing errors rather than on their accumulation and possible canceling. Such experiments are performed with the Max-min flow-level model in [38] as well as with the GTNetS packet-level simulator. The *mean error* and the *max error* of the flow-level model for one experiment is computed as (see [38] for the rationale for using this logarithmic measure):

$$MeanLogErr = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \left| \log\left(\varrho_f^{flow}\right) - \log\left(\varrho_f^{packet}\right) \right|$$

$$MaxLogErr = \max_{f \in \mathcal{F}} \left| \log\left(\varrho_f^{flow}\right) - \log\left(\varrho_f^{packet}\right) \right|$$

[1] All scenarios, simulation traces and analysis scripts are available at http://simgrid.gforge.inria.fr/network_validation/.

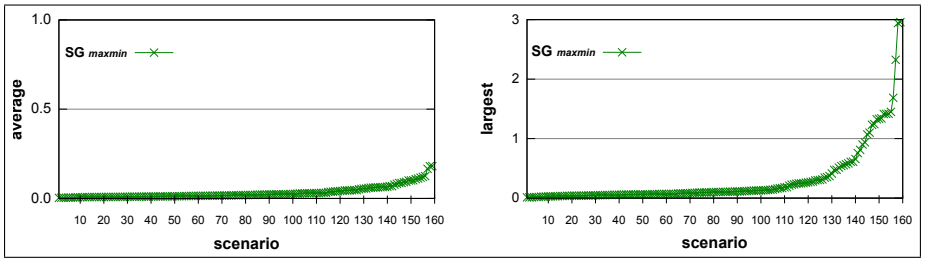


Figure 2: Mean logarithmic error (left) and max logarithmic error (right) for all experimental scenarios for the flow-level model in [38].

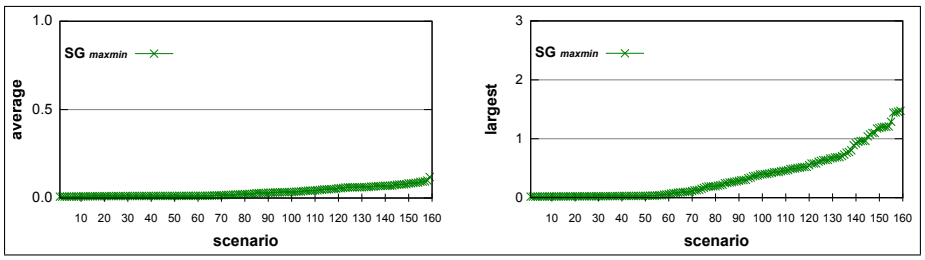


Figure 3: Mean logarithmic error (left) and max logarithmic error (right) for all experimental scenarios for the flow-level model in [38], after removing *phase effects* by configuring GTNetS with the RED queue policy.

Figure 2 depicts obtained result for the mean error (left graph) and the max error (right graph). The vertical axis shows the error, and the horizontal axis corresponds to the different experimental scenarios, sorted by increasing error. The main observation is that while some scenarios have low mean and max error, some lead to egregious errors. In particular, the max error reaches a value close to 3. Consequently, over all experiments, there is at least one flow whose bandwidth is a $e^3 \approx 20$ factor away from the bandwidth obtained using packet-level simulation. The negative conclusion is that simulation results obtained with flow-level simulation are "mostly reasonable," with casts doubts about how such simulation could be used meaningfully.

In what follows we investigate the source of the large observed error. A cursory exploration of the results does not reveal any pattern and over or under estimation errors seem random. Instead of seeking a pattern in the existing results, we pick a scenario in which at least one of the flows has large error, and iteratively remove flows that did not lead to significant error. We obtain a simplified experimental scenario (only two flows and a few links) that still leads to large errors. Finding that the source of the error remains mysterious even in such a simple case, we attempt to round up platform parameters (latency and bandwidth) to integral values to do a by-hand analytical evaluation of the flow-level bandwidth sharing model. Surprisingly, the results from packet-level simulation after this rounding off change dramatically. In fact, in our experiments it can easily be shown that packet-level simulation is extremely sensitive to platform parameters. For instance, a slightly modified latency can lead to a radically different bandwidth sharing among two competing flows, which cannot be captured by a (continuous) flow-level model. It turns out that we have “rediscovered” a phenomenon called *phase effect* [15], which is due to the default Droptail router queue policy algorithm. Phase effects are unlikely in actual wide-area networks because of frequent

small burst flows that randomly break the Droptail queue pattern. But, an *artifact* of deterministic packet-level simulation is that it makes phase effects significantly more likely. Fortunately, phase effects do not occur in current networks because the most commonly deployed router queue policy today is RED. This policy does not behave deterministically, and in fact phase effects were one of the motivations for developing RED.

We can now revisit the results in Figure 2 after configuring GTNetS so that the RED policy is used instead of Droptail. The flow-level model needs to be re-instantiated as well, meaning that the α, β, and γ parameters described in Section 3.3 must be re-estimated based on new packet-level simulation results obtained for elemental experimental settings. The resulting values are: $\alpha = 13.01$, $\beta = 0.97$, and γ = 20537.14. Newly obtained results for the 160 experimental scenarios are shown in Figure 3. The mean error decreases marginally. More important, the max error is greatly improved from close to 3 (a factor $e^3 \approx 20$) to a value below 1.6 (a factor lower than $e^{1.6} \approx 5$). The very large errors that remained unexplained in [38] were indeed caused by phase effects. While a factor 5 is better than a factor 20, there is still much room for improvement. Furthermore, comparing the graph on the right-hand side of Figure 2 to the graph on the right-hand side of Figure 3, we see that the max error is more "balanced" over the experiments (e.g., the max error of the 100-th scenario in the former is lower than that of the 100-th scenario in the latter).

4.2 Picking a critical workload

Following the critical method, one should evaluate a model over cases in which it performs poorly. While some of the experimental scenarios used in the previous section lead to substantial errors, many lead to low mean error, and to a lesser extent to low max error. Consequently, many of these scenarios fall into the "accumulating cases in which the model is effective" category instead of being "crucial experiments." Inspecting the scenarios in detail, we notice that most of the flows are limited by their RTTs. Such flows are good cases for a flow-level model because constraints 5 essentially bypass the core of the bandwidth sharing formulation. This observation is confirmed in Figure 4. The top part of the figure plots the max error over all experiments, sorted by increasing error. The bottom part, which uses the same experiment order, plots the percentage of flows that are RTT-bound. We see a clear inverse correlation between the max error and the fraction of flows that are RTT-bound.

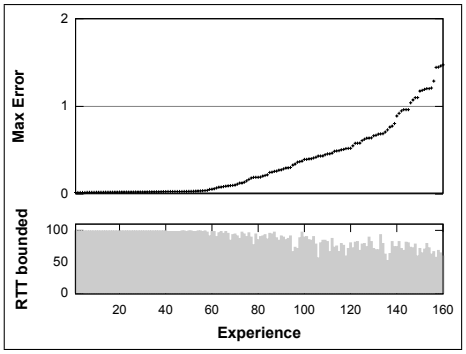


Figure 4: Max error and the percentage of RTT-bound flows for all experiments, sorted along the horizontal axis by increasing max error.

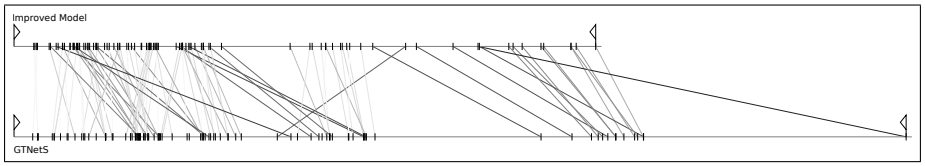


Figure 5: Comparison of flow completion times with flow-level simulation (top timeline) and packet-level simulation (bottom timeline); lines connecting timeline markers correspond to the same flow in both simulations and are darker for larger completion time mismatches.

Recall that network topologies for the experimental scenarios are generated using the Waxman model. This model is likely to create several possible paths between two sets of end-points (Figure 1(a) shows an example of this type of topology). The number of flows contending for bandwidth on the same link may be low, meaning that most flows could be RTT-bound. Consequently, the Waxman model, although well recognized, is likely not appropriate for generating crucial experiments when evaluating bandwidth sharing models. By contrast, Figure 1(b) shows an example of topology created with the Tiers algorithm [12], which uses a three-step space-based hierarchical approach. The resulting topologies are hierarchical and have thus both global (in the core of the network) and local (on the edges of the network) bottleneck links. We opt for using Tiers as a way to generate a new set of experimental scenarios that aim at further invalidating the flow-level model. Figure 10 shows that the mean error with the new topologies become larger than 1.25 (compared to below 0.5 with the Waxman topologies), and the max error is now above 6 (compared to below 2 with the Waxman topologies). It is with this new set of experimental scenarios that we continue our (in)validation study of our flow-level model in the next section.

4.3 Cross traffic effects

Our new experimental scenarios have larger errors, making "bad cases" easier to identify. To understanding the root causes of the errors we use a multi-scale, interactive and exploratory visualization tool called Triva [33, 34, 36]. We select some of the scenarios that have large maximum error and run them until completion of all flows so that we can compare flow completion times and not only instantaneous bandwidth shares. Figure 5, which is created using Triva, shows two timelines, the top one corresponding to our flow-level model and the bottom one corresponding to packet-level simulation. Each timeline marker represents the completion time of a flow. Triva connects two markers on the two timelines by a line if they correspond to the completion of the same flow. In a no-error situation, all these lines would be vertical. For easier visualization, Triva uses gray tonalities for these lines: the darker the line to larger the error (i.e., the less vertical the line). It is thus straightforward to identify which flows have large errors without being "distracted" by low-error flows.

Two observations can be made from Figure 5. First, the flow-level timeline is shorter, indicating that the simulated time of all flows is smaller with flow-level simulation than with packet-level simulation (about 65% in this example). Second, some flows do have low errors but many of them have very large errors.

We now use a graph-based visualization, another Triva feature, which shows the bandwidth utilization for all networks links in the topology, paying close attention to those large-error flows identified in Figure 5. This visualization is shown in Figure 6.

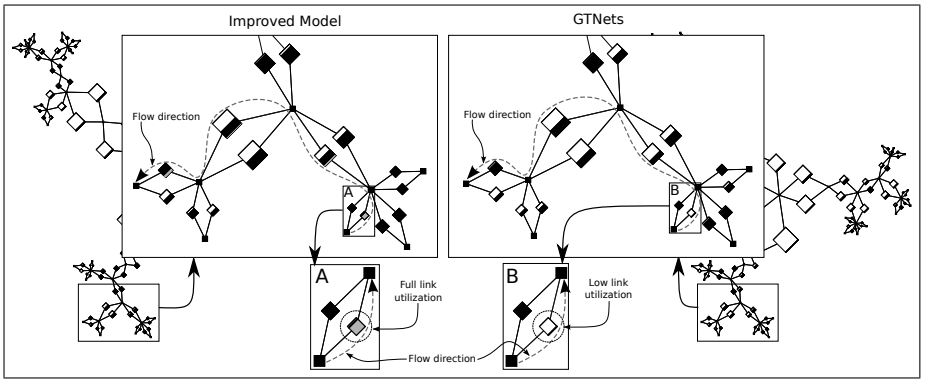


Figure 6: Visualization of network link utilization when all communication flows are active in the topology with a highlighted large-error flow.

Routers and hosts of the platform are represented by black squares, while network links are represented by diamonds. The size of a diamond is proportional to the link's bandwidth capacity. A dark color fill indicates link utilization (the more filled the diamond the more utilized the link). Utilization is computed as an average over a given time frame.

Figure 6 shows two visualizations, one for flow-level simulation (left) and one for packet-level simulation (right). Utilization values are averaged over the time from the onset of the simulation until the completion time of the first flow, thus ensuring that all flows are present in our visualization. We select a flow with large error, and show part of its network path in a zoomed region of the full topology. In the case of flow-level simulation, the network link that limits this flow, shown in the region labeled A, is, as expected, fully saturated. A gray fill in the figure shows the capacity allocated to our target flow. Surprisingly, in packet-level simulation a much lower bandwidth share is allocated to this flow and this network link, now in the region labeled B, is not saturated at all. In fact, our target flow receives an insignificant fraction of the bandwidth along the whole path whereas none of the links on the path are saturated. We conclude that TCP under-utilizes this network path and that a bandwidth sharing model based on a global optimization under the constraints in Eq. (1) has no chance of ever capturing this behavior. The same observation can be made for other flows in this experimental scenario, over other time frames, and in other experimental scenarios. Under-utilization of network links by TCP is thus a key reason that contributes to the large errors seen in our experimental scenarios: our flow-level model gives "too much" bandwidth to flows.

In a view to understanding the cause of the error we remove all the flows that do not use the link that showed widely different utilization in flow-level and packet-level simulations. The simplified experimental scenario still suffers from the same discrepancy between the two simulation, thus suggesting the following two falsifying laws:

1. On a single full-duplex link with capacity B, the bandwidth allocated to n flows is approximately B/n.

2. On a single full-duplex link with capacity B, the bandwidth allocated to flows going in a direction is independent from that allocated to flows going in the reverse direction.

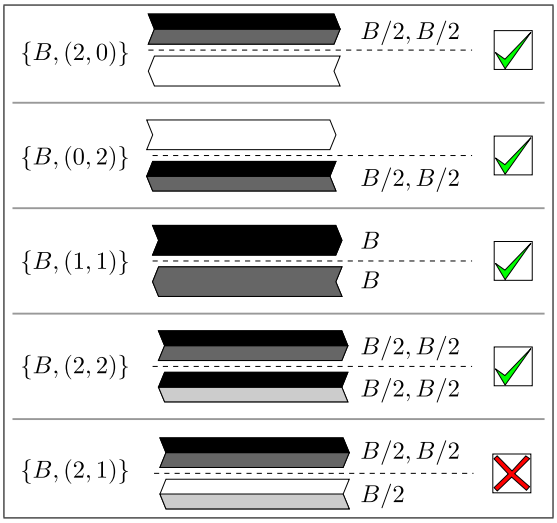
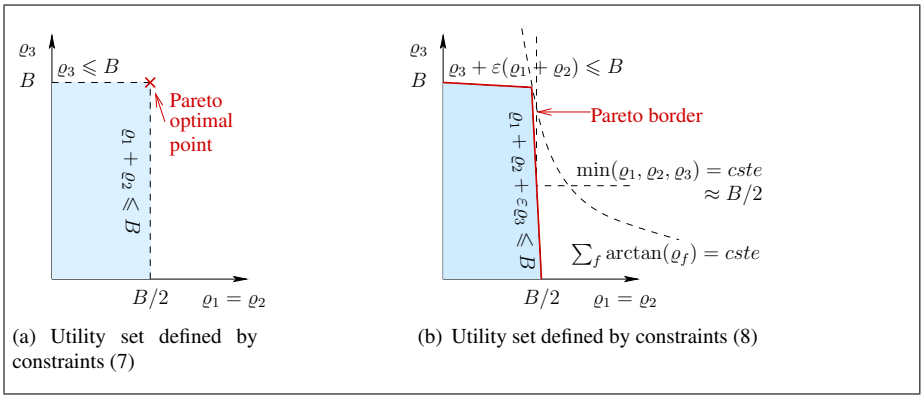


Figure 7: Five cross traffic interference scenarios: $\{B, (n, p)\}$ denotes a scenario with one link of capacity B, with n flows in one direction (shown above the dashed lines) and p flows in the reverse direction (shown below the dashed lines). The bandwidths allocated to the flows are shown on the right-hand side of each scenario. The last scenario shows an asymmetric situation.

We conducted packet-level simulations to test these two laws. Let us denote by $\{B, (n, p)\}$ a scenario with a single full-duplex bottleneck link of capacity B, n flows going through the link in one direction, and p flows going through the link in the reverse direction. We denote the outcome of the scenario by (B_1, B_2), where B_1 is the bandwidth allocated to the n flows and B_2 is the bandwidth allocated to the p reverse flows (in all cases all flows in the same direction are allocated the same bandwidth).

Figure 7 depicts five example scenarios, showing approximate results achieved in simulation. The bandwidth allocation for $\{B, (n, 0)\}$ (resp. $\{B, (0, p)\}$) is always approximately $(B/n, 0)$ (resp. $(0, B/p)$). Since the network link is full-duplex, expectedly simulations also show that $\{B, (1, 1)\}$ leads to the allocation (B, B), which seems to validate the second falsifying law above since two reverse flows can make full usage of the full-duplex bandwidth. Likewise, $\{B, (2, 2)\}$ leads to $(B/2, B/2)$. Surprisingly though, $\{B, (2, 1)\}$ does not lead to $(B/2, B)$ as one would expect but instead to $(B/2, B/2)$. More generally, our experiments show that $\{B, (n, p)\}$ leads to allocation $(B/\max(n, p), B/\max(n, p))$. Given how surprising this result seems, one may wonder whether it is not a simulation artifact. We conducted dozens of experiments on real-world networks, with host pairs connected via a direct full-duplex link and host pairs connected via a sequence of full-duplex links. The phenomenon above was confirmed in every tested configuration. Using network monitoring tools, namely `tcpdump` and `wireshark`, we were able to understand that link under utilization is due to the *interference* between acknowledgments for the traffic in one direction and the traffic in the other direction, which we term cross traffic. Acknowledgment packets are queued with data packets from the traffic, thus slowing down the cross traffic. This phenomenon, known as *ACK compression*, is very common for ADSL connections. In perfectly symmetrical cases, although *ACK compression* may still occur, delayed ACKs should make it disappear. As recently observed in [17], our observed poor link utilization is more likely to be explained by a *data pendulum* effect where data and



(a) Utility set defined by constraints (7)

(b) Utility set defined by constraints (8)

Figure 8: Consequence of utility set deformation

ACK segments alternatively fill only one of the link buffers. At any rate, such phenomena, which is not modeled by any previously proposed flow-level model, results in seemingly underutilized bottleneck links in flow-level simulations.

4.4 Accounting for cross traffic

It turns out that, under constraints (1), our flow-level model is inherently incapable of capturing the cross-traffic effect identified in the previous section. To illustrate this, consider a scenario with two hosts M_1 and M_2 connected via one link of capacity B. If we have two flows from M_1 to M_2 and one flow from M_2 to M_1, constraints (1) are rewritten as:

$$\begin{cases} \varrho_1 + \varrho_2 \leqslant B \\ \varrho_3 \leqslant B \end{cases} \quad \text{and (by symmetry) } \varrho_1 = \varrho_2 \tag{7}$$

Figure 8(a) depicts the corresponding utility set, in which there is a single Pareto optimal point $(\varrho_1 = \varrho_2, \varrho_3) = (B/2, B)$. This point is far from the actual bandwidth share achieved by TCP due to the cross-traffic effect. And yet, formulating bandwidth sharing as an optimization problem always produces a Pareto-optimal solution. Consequently, such flow-level models cannot account for cross-traffic effects with constraints (1). We propose to change these constraints as follows:

$$\forall l \in \mathcal{L}, \sum_{f \text{ going through } l} \varrho_f + \varepsilon \cdot \left(\sum_{f\text{'s ack through } l} \varrho_f \right) \leqslant B_l \ . \tag{8}$$

Figure 8(b) depicts the utility set defined by these new constraints. The utility set is deformed so that the unique Pareto-optimal point has become a whole Pareto border. The points of this border can be reached using Max-min optimization, thus allowing for a valid bandwidth share to be computed.

It is interesting to note that the flow-level model proposed by Low [22] here remains ineffective because of the inherent shape of its objective function. The shape of the isolines of sum-based objective functions is such that, unless the utility set is heavily distorted, the optimal solution is always the rightmost and uppermost vertex, which we have seen is far from the bandwidth share achieved by TCP. More generally, the absence of interference between acknowledgments and data traffic is one of the

main assumptions used to build bottom-up flow-level models (i.e., making the analogy between the equations governing expected window size and a distributed gradient algorithm that optimizes a sum-based function). [22] defines "d_f*, the equilibrium round trip time (propagation plus equilibrium queuing delay) of flow* f*, which is assumed to be constant.*" In the previous example, however, d_f is *not* constant and actually depends on the throughput of the other flows, which influence the queuing delay. Such models are thus unlikely to be valid as soon as there is cross-traffic, which drastically limits their usefulness. This issue is not encountered and thus not identified in any of the experiments in [23, 22, 24] because all flows are along the same direction in those experiments.

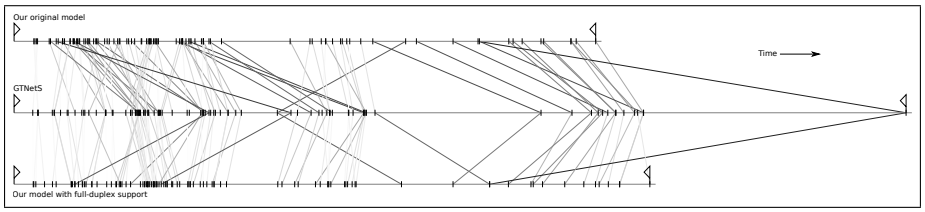


Figure 9: Comparison of flow completion times with the original flow-level simulation (top timeline); packet-level simulation (middle timeline); and the improved flow-level simulation with modified constraints to account for cross-traffic (bottom timeline).

Returning to our Max-min model, we evaluate the effect of the modified constraints, returning to the experimental scenario investigated at the beginning of the previous section. Figure 9 is similar to Figure 5, but also displays the timeline for the improved model at the bottom. It can plainly be seen that there are fewer dark lines to the timeline at the bottom than to the timeline at the top. The modified constrains thus improve the quality of our flow-level model.

Figure 10 shows the benefit of the modified model over the original model for the entire set of critical experimental scenarios, with scenarios sorted by increasing error of the original model. We see that the mean error is always improved by the modified model, while most max errors are also improved. Importantly, the largest max error with the modified model is below 4, while it can be higher than 6 with the original model.

To illustrate the inability of arctan-based flow-level models to account for cross-traffic, Figure 11 shows results for our improved Max-min model and the model from [22].

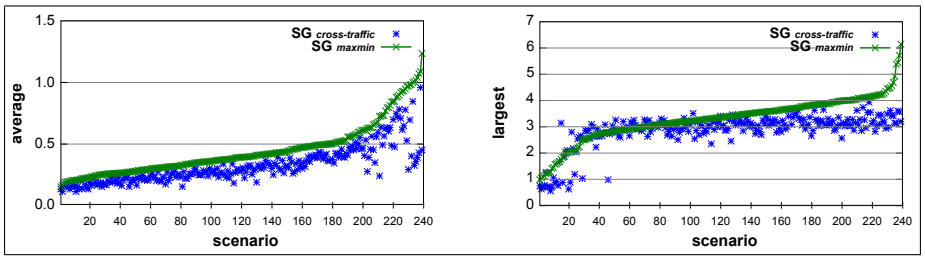


Figure 10: Mean logarithmic error (left) and max logarithmic error (right) for all experimental scenarios for the flow-level model SG_{maxmin} in [38] and the improved cross-traffic model $SG_{cross\text{-}traffic}$ proposed in Section 4.3, evaluated with topologies generated by Tiers.

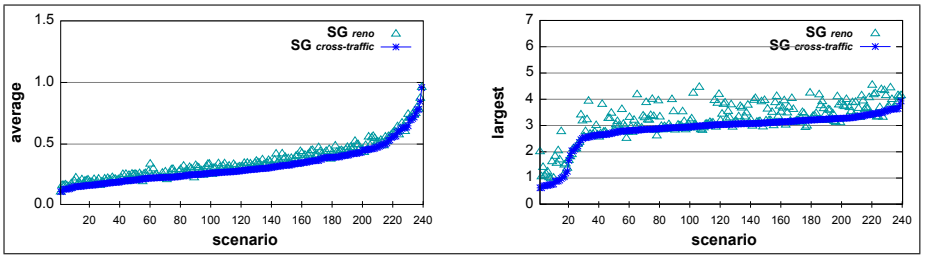


Figure 11: Mean logarithmic error (left) and max logarithmic error (right) for all experimental scenarios for the cross-traffic aware model based on Max-min, $SG_{cross\text{-}traffic}$, and the arctan-based model in [22], SG_{reno}.

Experimental scenarios are sorted by increasing error of the Max-min model. Although the model from [22] is attractive because built bottom-up from first principles of the TCP algorithms, its mean error is almost always slightly larger than that obtained with our improved top-down Max-min model. More importantly perhaps, the Max-min model leads to significantly lower max error, i.e., better worst-case behavior. We conclude that accounting for cross-traffic effects is paramount when developing a flow-level model.

5 Limits of flow-level modeling?

Account for cross-traffic effects has improved our flow-level model substantially in terms of mean error over a range of crucial experimental scenarios. However, when considering flow completion times rather than bandwidth shares when all flows are active, many worst-case scenarios show no improvement at all. These flows complete much later in packet-level simulation than in flow-level simulation, and the cause of these errors is not the cross-traffic effect. Using our visualization tool we attempted to find a topological pattern that could be shared among all these worst-case scenarios, but were not successful. We also tried the traditional approach of simplifying scenarios by iteratively removing flows to isolate those with the worst errors. Such simplifications were also unsuccessful as errors always disappeared early in the iterative process.

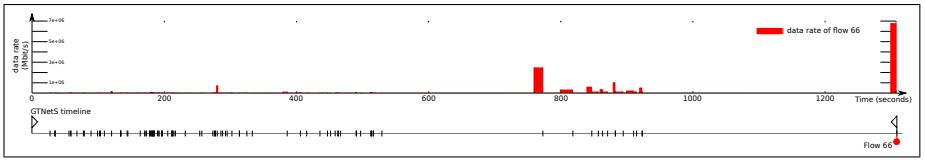


Figure 12: Evolution of the throughput of the last finishing flow, which has large error. Because of high contention, this flow stalls 65% of the time. When there is no other remaining connection in the network, it does not transmit a single byte for 380 seconds, and finally completes the transfer in a few seconds.

Eventually, we isolated "unexpected" behavior in the packet-level simulations, e.g., for the last flow to complete in the packet-level simulation timeline in Figure 9. If this flow were the only flow, packet level simulation shows that it would complete in less than 4 seconds. And yet, in the full simulation, the next-to-last flow finishes 390 seconds before the completion of this last flow! Analyzing the communication trace

from the simulation shows a long period of inactivity for this flow. Figure 12 shows the data transfer rate of the last flow vs. time. We see that this flow receives some bandwidth in the beginning of the simulation, then stalls for 380 seconds, and then completes rapidly at the end of the simulation, receiving the full bandwidth of all the network links on its path. These results are with the GTNetS packet-level simulator, but we have observed the same behavior with NS3, thus indicating that this phenomenon is likely not a simulation artifact. In fact, the long inactivity period is due to TCP timeouts that occur for highly contended flows. We surmise that such effects may be impossible to model using a (continuous) flow-level model. A simulation of the state of the TCP stack would certainly be required, thus blurring the line between flow-level and packet-level simulation.

Although it may happen that some flows stall for long periods of time, situations as extreme as the ones we explored in our critical experimental scenarios may be unlikely in practice. This is somewhat comforting, and the quality of our improved Max-min model is high for large messages in every situations we have explored, except for the most contended ones (i.e., with large latencies and low bandwidth capacities). In these cases the high error is due to the discrete nature of the TCP protocol, which, by design, is not captured by flow-level models.

6 Conclusion

When simulating large distributed systems and applications, the use of packet-level simulation for network communications, albeit widely accepted to be accurate, typically proves unscalable. An alternative is to use simulation that relies on flow-level models of network communications. Several flow-level models of TCP have been proposed and used in the literature but, perhaps surprisingly, few works have thoroughly investigated their validity. On such work is our previous study in [38]. Although the core of a flow-level model is the bandwidth sharing model, that study showed that bandwidth sharing was not the primary source of model errors: errors were due primarily to poor instantiating of the model parameters. The final evaluation in [38] showed that a model based on Max-min fairness leads to good results on average even in complex scenarios. However, some flows suffered from very large errors, which remained unexplained. Generally speaking, as seen in Section 2, it is never possible to establish the validity of an empirical model entirely. Instead, one must seek to invalidate the model as a way to ascertain the amount of confidence that can be placed into it, and perhaps in the process propose improvements. In this article, we have followed such an invalidation path.

Our first source or error comes from our reference packet-level simulator, which used the Droptail router queue policy. This policy creates simulation artifacts known as phase effects, which compromise the results in [38]. When removing these artifacts by using the RED queue policy, errors are decreased overall. It is interesting to note that this issue was discovered due to our implicit trust into our flow-level model, but that this trust was somewhat misplaced as large errors remain. While looking for a second source of error, we determined that our benchmark workload, although seemingly standard, led to low-contention experimental scenarios. These scenarios are easy cases for a flow-level model that relies on solving an optimization problem. We thus generated a new set of experimental scenarios, and expectedly, modeling errors were vastly increased. Actively seeking scenarios in which the error of the model proposed by the authors is as large as possible is perhaps atypical in the literature. Nevertheless,

it is the course of action dictated by the critical method described in Section 2.

Turning our attention to the sources of these enlarged errors, we eventually discovered that many were due to cross-traffic interference. The reason is that the data transfer rate of a TCP flow is strongly dependent upon the rate at which acknowledgment packets arrive, which is easily demonstrated to be influenced by flows going in the reverse direction both in packet-level simulation and in real-world experiments. Perhaps surprisingly, it is straightforward to modify our Max-min-based flow-level model to account for this cross-traffic effect. By contrast, such modifications seem out of reach for the flow-level models proposed by others in the literature. In fact, these models were built in a bottom-up fashion, completely ignoring cross-traffic effects. Furthermore, their "validation" had been done by illustrating their accuracy on a few simple cases. It is likely that another model (like our Max-min-based model) would have produce the same results in such simple settings, which shows the inherent limitation of such methodology. To the best of our knowledge, our Max-min based model augmented with cross-traffic support is the best flow-level model of TCP available to date.

Our results show that this model leads to high-quality results in a wide range of settings. However, it is far from being perfect. We have found situations in which the bandwidth allotted to a flow according to the model is very different from the bandwidth allotted to it according to packet-level simulation. A detailed study of these results shows that flow-level models have little hope to model such situations correctly. Yet, these situations are extreme (high contention on links with very low capacity) and thus perhaps unlikely in practice.

In a sense, our investigation of flow-level models has reached a dead-end because all erroneous behaviors that remain can be imputed to violating the steady-state assumption that is the central tenet of all flow-level models. A future direction would consist in providing sufficient conditions for the steady-state assumption to be invalid. This would allow to better identify the validity domain of flow-level models and allow further investigation of invalidation scenarios. Ultimately, building on such results, a simulator relying on flow-level models should warn its users when it steps outside the validity domain of its simulation models.

References

[1] A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 59:509–512, 1999.

[2] W. H. Bell, D. G. Cameron, A. P. Millar, L. Capozza, K. Stockinger, and F. Zini. OptorSim: A grid simulator for studying dynamic data replication strategies. *International Journal of High Performance Computing and Applications*, 17(4), 2003.

[3] D. P. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1992.

[4] R. Buyya and M. Murshed. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, 14(13), Nov. 2002.

[5] H. Casanova. SimGrid: A toolkit for the simulation of application scheduling. In *First IEEE International Symposium on Cluster Computing and the Grid (CCGrid'01)*, Brisbane, Australia, May 2001.

[6] H. Casanova, A. Legrand, and L. Marchal. Scheduling distributed applications: the SimGrid simulation framework. In *Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)*. IEEE Computer Society Press, May 2003.

[7] H. Casanova, A. Legrand, and M. Quinson. SimGrid: a generic framework for large-scale distributed experiments. In *Proceedings of the 10th Conference on Computer Modeling and Simulation (EuroSim'08)*, 2008.

[8] H. Casanova and L. Marchal. A network model for simulation of grid application. Technical Report 2002-40, LIP, Oct. 2002.

[9] Q. Chen, H. Chang, R. Govindan, and S. Jamin. The origin of power laws in Internet topologies revisited. In *INFOCOM*, pages 608–617, 2002.

[10] D. N. Chiu. Some observations on fairness of bandwidth sharing. Technical report, Sun Microsystems, 1999.

[11] P.-N. Clauss, M. Stillwell, S. Genaud, F. Suter, H. Casanova, and M. Quinson. Single node on-line simulation of MPI applications with smpi. In *25th IEEE International Parallel and Distributed Processing Symposium (IPDPS'11)*, Anchorage (Alaska) USA, May 2011.

[12] M. B. Doar. A better model for generating test networks. In *IEEE GLOBECOM*, pages 86–93, Nov. 1996.

[13] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[14] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[15] S. Floyd and V. Jacobson. On traffic phase effects in packet-switched gateways. *Internetworking: Research and Experience*, 3:115–156, 1992.

[16] K. Fujiwara and H. Casanova. Speed and accuracy of network simulation in the simgrid framework. In *Proceedings of the 2nd international conference on performance evaluation methodologies and tools*, pages 1–10, 2007.

[17] M. Heusse, S. A. Merritt, T. X. Brown, and A. Duda. Two-way TCP Connections: Old Problem, New Insight. *ACM Computer Communication Review*, Apr. 2011.

[18] T. Issariyakul and E. Hossain. *Introduction to Network Simulator NS2*. Springer Link, July 2008.

[19] M. Jain, R. S. Prasad, and C. Dovrolis. The TCP bandwidth-delay product revisited: network buffering, cross traffic, and socket buffer auto-sizing. Technical Report GIT-CERCS-03-02, Georgia Institute of Technology, 2003.

[20] S. Jansen and A. McGregor. Validation of simulated real world TCP stacks. In *Winter Simulation Conference*, 2007.

[21] A. Lakhina, J. Byers, M. Crovella, and P. Xie. Sampling biases in ip topology measurements. In *INFOCOM*, 2003.

[22] S. H. Low. A duality model of TCP and queue management algorithms. *IEEE/ACM Transactions on Networking*, 11(4), 2003.

[23] S. H. Low, L. L. Peterson, and L. Wang. Understanding vegas: a duality model. *Journal of the ACM*, 49(2), Mar. 2002.

[24] S. H. Low and R. Srikant. A mathematical framework for designing a low-loss, low-delay internet. *Network and Spatial Economics*, 4:75–102, 2004.

[25] G. Marfia, C. Palazzi, G. Pau, M. Gerla, M. Y. Sanadidi, and M. Roccetti. TCP-Libra: exploring RTT fairness for TCP. Technical Report 050037, UCLA Computer Science Department, 2005.

[26] M. Mathis, J. Semke, and J. Mahdavi. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3), 1997.

[27] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An approach to universal topology generation. In *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS'01*, Cincinnati, Ohio, Aug. 2001.

[28] NS3. The Network Simulator 3: http://www.nsnam.org/, Last visited in November 2011.

[29] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer. GroudSim: An Event-based Simulation Framework for Computational Grids and Clouds. In *CoreGRID/ERCIM Workshop on Grids, Clouds and P2P Computing*. Springer, 2010.

[30] K. Popper. *Objective Knowledge: An Evolutionary Approach*. Oxford University Press, 1972.

[31] K. Ranganathan and I. Foster. Decoupling computation and data scheduling in distributed data-intensive applications. In *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, pages 352–358. IEEE, 2002.

[32] G. F. Riley. The georgia tech network simulator. In *ACM SIGCOMM workshop on Models, Methods and Tools for Reproducible Network Research*, pages 5–12, Karlsruhe, Germany, 2003.

[33] L. Schnorr, A. Legrand, and J.-M. Vincent. Detection and analysis of resource usage anomalies in large distributed systems through multi-scale visualization. *Concurrency and Computation: Practice and Experience.*, 2011.

[34] L. M. Schnorr, G. Huard, and P. O. A. Navaux. Triva: Interactive 3D visualization for performance analysis of parallel applications. *Future Generation Computer Systems*, 26(3):348–358, 2010.

[35] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topology generators: Degree-based vs structural. In *SIGCOMM*, Aug. 2002.

[36] Triva. http://triva.gforge.inria.fr, Last visited in November 2011.

[37] A. Varga and R. Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Simutools '08, pages 60:1–60:10, 2008.

[38] P. Velho and A. Legrand. Accuracy study and improvement of network simulation in the simgrid framework. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, 2009.

[39] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, Dec. 1988.

[40] G. Zheng, G. Kakulapati, and L. V. Kale. BigSim: A parallel simulator for performance prediction of extremely large parallel machines. In *IPDPS*, 2004.

[41] G. Zheng, T. Wilmarth, O. S. Lawlor, L. V. Kalé, S. Adve, and D. Padua. Performance modeling and programming environments for petaflops computers and the blue gene machine. In *Computer Science*. IEEE Press, 2004.

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399